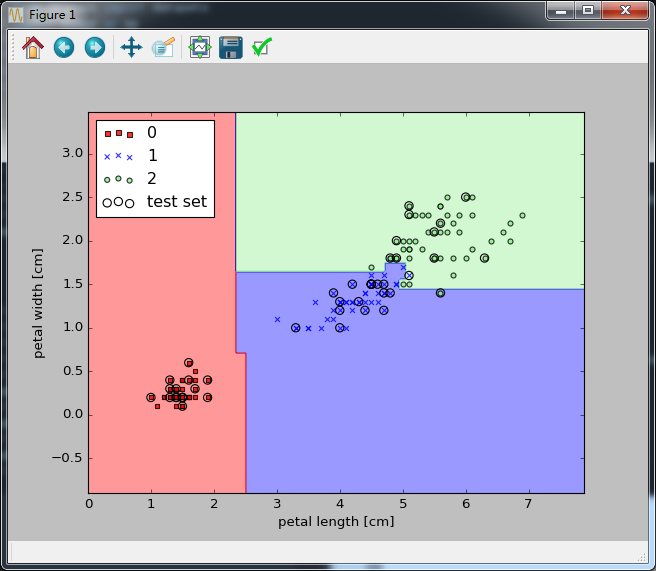
**Random forests**

A random forest can be considered as an ensemble of decision trees. The idea behind ensemble learning is to combine weak learners to build a more robust model, a strong learner, that has a better generalization error and is less susceptible to overftting. The random forest algorithm can be summarized in four simple steps:

1. Draw a random **bootstrap** sample of size n (randomly choose n samples from the training set with replacement).
2. Grow a decision tree from the bootstrap sample. At each node:
3. Randomly select d features without replacement;
4. Split the node using the feature that provides the best split according to the objective function, for instance, by maximizing the information gain.
5. Repeat the step 1 to 2 k times.
6. Aggregate the prediction by each tree to assign the class label by **majority vote**.

There is a slight modifcation in step 2 when we are training the individual decision trees: instead of evaluating all features to determine the best split at each node, we only consider a random subset of those.

Although random forests don't offer the same level of interpretability as decision trees, a big advantage of random forests is that we don't have to worry so much about choosing good hyperparameter values. We typically don't need to prune the random forest since the ensemble model is quite robust to noise from the individual decision trees. The only parameter that we really need to care about in practice is the number of trees *k* (step 3) that we choose for the random forest. Typically, the larger the number of trees, the better the performance of the random forest classifer at the expense of an increased computational cost.



Reference:

Python Machine learning,book